

CLAIMS

1. A method to detect a fault in a CPU of an industrial controller during on-line safety control of real world objects comprising the steps of
 - compiling an application program into assembler instructions, which application program was previously defined in a high level language intended for safety control,
- 10 **characterized** by that the method comprising the steps of
 - compiling a test application into assembler instructions where the assembler instructions is a subset of the total number of assembler instructions available for the CPU, which test application was previously defined in said high level language intended for safety control and the test application covers at least all language constructs used in the application program,
 - downloading the application program and the test application to a central unit of an industrial controller,
 - executing repeatedly the test application in the industrial controller,
 - comparing repeatedly by means of a test module a result from the test application with the pre-defined result in the test module,
 - detecting a fault in the CPU as the result from the test application does not equal the pre-defined result stored in the test module and the unexpected result of the test application is due to the execution of an assembler instruction of the test application,
 - aborting the execution of the application program wherein the application program is prohibited from executing the assembler instruction which otherwise would cause the application program to fail.
- 35 2. A method according to claim 1 where the assembler version of the test application comprise assembler code derived from all language constructs in the high-level

language available for safety control of real world objects.

3. A method according to claim 1 or claim 2 where the
5 high level language intended for safety control is based
on IEC 61131-3.

4. A method according to claim 3, **characterized** in that
the step of defining a test application comprise an
10 analyses of the application in order to determine subset
and software libraries used in the said application code.

5. A method according to claim 4, **characterized** in that
the step of defining a test application is made
15 automatically without any additional command from an
application programmer.

6. A method according to claim 5, **characterized** in that
the step of executing the test application repeatedly is
20 performed by a cyclic execution of the test application
where the cycle time is determined from a given process
safety time value.

7. A method according to claim 6, **characterized** in that
25 the said test application before an execution receives a
set of input values and the input values are generated by
means of the test module.

8. A method according to claim 7, **characterized** in that
30 the down-loading step of application program and test
application comprise the additional step of down-loading
a predefined result.

10. A computer program product, for use in an industrial
35 control system, containing software code means loadable

into the central unit of an industrial controller intended for safety control of real world objects, said computer program product **characterized** in that it comprises means to make the industrial controller:

- 5 - execute repeatedly the test application in the industrial controller,
- compare repeatedly by means of a test module a result from the test application with the pre-defined result in the test module,
- 10 - detect a fault in the CPU as the result from the test application does not equal the pre-defined result stored in the test module and the unexpected result of the test application is due to the execution of an assembler instruction of the test application,
- 15 - abort the execution of the application program wherein the application program is prohibited from executing the assembler instruction which otherwise would cause the application program to fail, all steps according to the method in claim 1.

20

- 11. An industrial control system, comprising an industrial controller with a central unit equipped with a CPU intended for safety control of real world objects, an I/O system **characterized** in that the CPU is subject to
- 25 fault detection according to the method in claim 1.